Mitigating the Impact of Fake News

A Major Qualifying Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the Degree of Bachelor of Science
in
Computer Science

TEAM MEMBERS
Alexander Rus
Alex Tian

ADVISOR:
Professor Kyumin Lee

Abstract

Fake news in social media is a pressing issue. The goal of this project was to mitigate the impact of fake news by developing recommendation systems that can recommend fact-checking URLs to “guardians”. We created three recommendation systems, improved the GAU model, determined the best recommendation system, and developed a web application that facilitates use of the recommendation systems.

Acknowledgements

We would like to thank WPI Professor Kyumin Lee and Nguyen Vo for assisting us throughout the project by answering our questions, providing us with valuable resources, and keeping us on track. Their feedback and assistance helped us to shape our project into its current form.

Executive Summary

Social media is a popular distributor of news for many people around the world. The global average penetration rate of social media in January 2020 was 49% (Clement, 2020). In addition, 68% of American adults said they at least occasionally got news on social media in 2018, and 20% said they got news there often (Shearer & Matsa, 2018). Unfortunately, a pressing issue with news distributed with social media is the prevalence of fake news, which is news that is deliberately inaccurate. Leading up to the 2016 US Election, 6.6 million tweets with links to fake news publishers were found (Hindman & Barash, 2018). One method of combating fake news is fact-checking, which is the act of determining the correctness of a statement such as a news article (Hameleers & van der Meer 2019). To combat fake news with fact-checking, we plan on using "guardians" (Vo & Lee, 2019). Guardians are Twitter users that show interest in correcting fake news by replying to tweets with fact-checking URLs or retweeting the reply made by a guardian. The GAU model was developed by Nguyen Vo and Kyumin Lee in 2019 to provide personalized recommendations of fact-checking URLs to guardians so that they can then use them in their retweets.

Goals and Objectives

Our goal was developing recommendation systems that can recommend fact-checking websites to guardians and building a web application that facilitates use of the recommendation systems.

We completed three objectives to achieve our goal.

1. First, we compared the results of User-Based, Item-Based, Matrix Factorization, and GAU recommendation systems;
2. Next, we improved the GAU model by tuning its parameters.
3. Third, we built a web application to facilitate use of our recommendation systems.

Methodology

First, we created the user-based and item-based recommendation systems, which utilized user-based collaborative filtering and item-based collaborative filtering respectively and are memory based techniques. Next, we built a matrix factorization recommendation system, which employs a model-based approach to collaborative filtering. Then, we improved the GAU model by tuning its parameters. Finally, we developed a web application using the MERN development stack that allows guardians to easily access their recommendations from each of the recommendation systems.

Results

We were able to improve the GAU model after tuning its parameters. We saw a 14.5% improvement in the test nDCG @ k = 10 when comparing the results using the initial parameters and the improved parameters. The most significant changes to the parameters were changing the loss function from single pointwise square to hinge and altering the alpha_gau, beta_gau, and gamma_gau variables.

We compared all four of the recommendation systems using the test nDCG @ k = 10 and found that the improved GAU model produces the most relevant recommendations and is the primary recommendation system that should be used to recommend fact-checking URLs to guardians.

The web application is a fully operational interactive interface that allows visitors to view the recommended URLs for a given Twitter user. Simply entering a username will return the Twitter user information for that user, as well as the lists of recommended URLs using item based recommendation, user based recommendation, matrix factorization, and the GAU Model. Built using MongoDB, Express, Node.js, and React, the user can not only view the results of the various recommendation models, they can also learn about the logic behind each system, as well as WPI and the MQP process. The web application can be accessed by visiting https://floating-retreat-14391.herokuapp.com/ .

Future Work

The goal of our project and the GAU model is to recommend fact-checking URLs to guardians. However, there is still future work that must be done before that goal is achieved.

1. The web application can be improved to be more efficient and allow recommendation to happen dynamically rather than statically. Instead of providing the results from the users provided in the dataset, we could allow new users to have their recommended URLs calculated in real time.
2. The GAU model's parameters can be tuned further. We explored many combinations of parameters with the hinge loss function, but other loss functions, such as bpr, single

pointwise square, and adaptive hinge may produce better results when used in undiscovered combinations.

3. Improvements can be made to the GAU model, such as addressing cold-start issues and investigating deep learning.

Table of Contents

List of Figures

1 Introduction

Social media is a popular distributor of news for many people around the world. According to the Pew Research Center, 68% of American adults said they at least occasionally got news on social media in 2018, and 20% said they got news there often (Shearer & Matsa, 2018). The most commonly used social media for news are Facebook, YouTube, and Twitter. However, 57% of these consumers believed the news they see on social media is largely inaccurate.

News that is deliberately inaccurate is called fake news. The prevalence of social media in today's society has allowed fake news to be created and spread easily (Soll, 2016). One method of combating fake news is fact-checking, which is the act of determining the correctness of a statement such as a news article. Fact-checking is done by news organizations, such as The Washington Post, and websites dedicated to fact-checking, such as Snopes.

Fact-checking has been shown to be effective at mitigating the effects of fake news. According to experiments done by Hameleers and van der Meer (2019), participants lowered agreement with attitudinally congruent political fake news and overcame political polarization after using fact-checkers. A problem with fact-checking is that people tend to avoid information that conflicts with their political beliefs (Stroud, 2007). Fact-checkers may not be read by people that hold incorrect beliefs.

The term “guardian” was coined by Nguyen Vo and Kyumin Lee in their research paper on the GAU model (2019). A guardian shows interest in correcting fake news by replying to tweets with fact-checking URLs or retweeting the reply made by a guardian. Direct guardians are guardians that reply to tweets, and secondary guardians retweet the replies. The GAU model was developed as a recommender system that can recommend fact-checking URLs to guardians.

The goal of our project was developing recommendation models that can recommend fact-checking websites to guardians and building a web application that facilitates use of the recommendation models. There were three objectives that helped us achieve our goal. First, we compared the results of our User-Based, Item-Based, Matrix Factorization, and GAU recommendation models. Second, we improved the GAU recommendation model. Finally, we built a web application to facilitate use of our recommendation models. We hope that our project is a step towards the increase of guardians using the GAU recommendation model and tweeting fact-checkers. We believe that guardians may be able to expose fact-checkers to those that would not have been exposed to them otherwise.

2 Background

This chapter begins with social media and its role in spreading news. Then we explain fake news and fact-checking. We conclude by describing how recommender systems work, introducing the various recommendation models we implemented, and explaining evaluation metrics of recommender systems.

2.1 Fake News

Fake news is news that is deliberately inaccurate. This section will cover news in social media and fact-checking.

2.1.1 News in Social Media

In the present times, social media is everywhere. The global average penetration rate of social media in January 2020 was 49% (Clement, 2020). With this knowledge in mind, it is important to note the amount of news consumed through social media. The Pew Research Center states that 68% of American adults at least occasionally got news on social media in 2018, and 20% said they got news there often (Shearer & Matsa, 2018).

In the month before the 2016 US election, Hindman and Barash (2018) found more than 6.6 million tweets with links to fake news publishers, showing the widespread problem of fake news on Twitter. Because so many people get their news on social media, it has become a platform for fake news to spread (Soll, 2016). In addition, news can be generated with algorithms or anyone

with Internet access, ignoring the fact-checking process of trained journalists and reporters that traditional news had. That is why it is important to help stop the effects of fake news in social media.

2.1.2 Fact-Checking

One method of combating fake news is fact-checking, which is the act of determining the correctness of a statement such as a news article. Fact-checking is done by news organizations, such as The Washington Post, and websites dedicated to fact-checking, such as Snopes. Fact-checking has been shown to be effective at mitigating the effects of fake news. According to experiments done by Hameleers and van der Meer (2019), participants lowered agreement with attitudinally congruent political fake news and overcame political polarization after using fact-checkers. A problem with fact-checking is that people tend to avoid information that conflicts with their political beliefs (Stroud, 2007). This means that fact-checkers may not be read by people that hold incorrect beliefs. To help solve this issue, there must be a way to show fact-checkers to these people.

2.2 Recommender Systems

Consumers today are faced with an overwhelming number of choices, and only a small portion of those choices are relevant to each user. To narrow down the choices to only the relevant ones, a recommender system can be used. A recommender system predicts the rating, either explicit or implicit, a user would give to an item (Ricci, Rokach, & Shapira, 2011). They are most commonly used to suggest products or some other good to users and consumers. An explicit

rating is one a user decides to give to a product, such as a rating from 1 to 5 stars. On the other hand, an implicit rating is given simply by deciding between using or not using a product. Recommender systems are implemented in many of the services people use every day, such as online shopping services like Amazon and streaming services such as Netflix, and are a crucial part of the consumer experience.

There are a variety of recommender system approaches (Liao, 2018). The approach we will be focusing on is collaborative filtering. This approach builds a model that assumes people will enjoy items similar to items they have previously enjoyed and will enjoy items that similar people enjoy (Grover, 2017). Generally, there are two techniques in collaborative filtering: memory based and model based. A memory based technique involves finding similar users or items and then taking a weighted average of ratings. A model based technique uses machine learning to train a model that can recommend items. All of this information is laid out in figure 2.1.

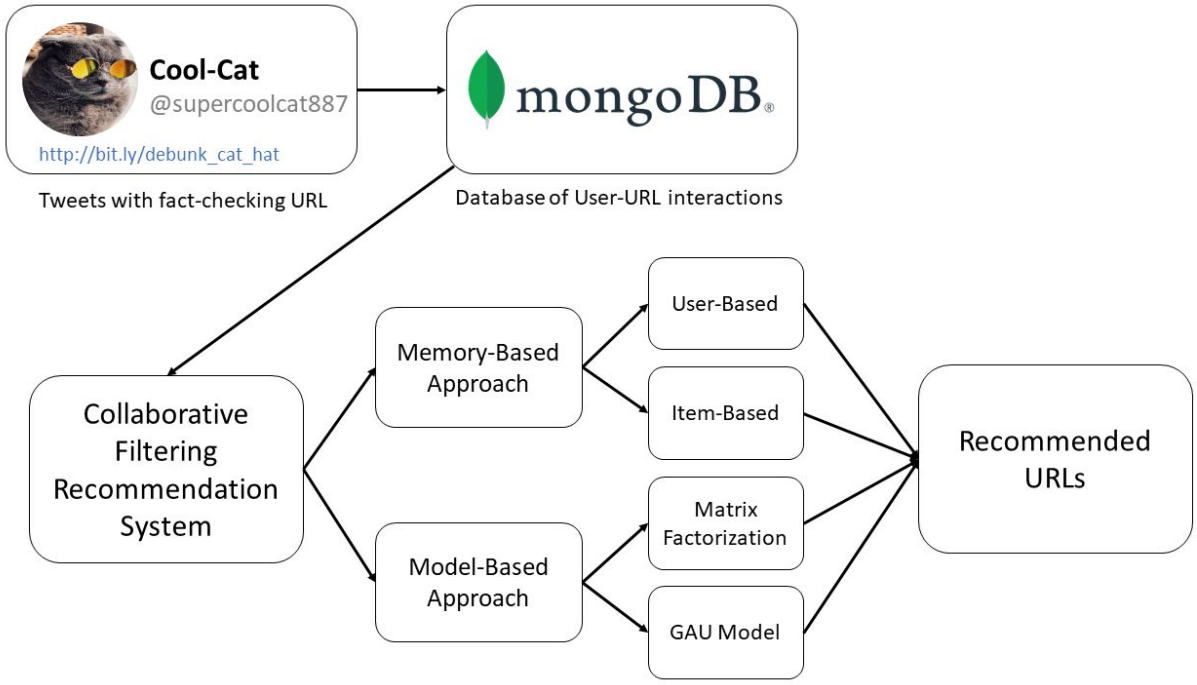

Figure 2.1: Recommender System Pipeline

2.2.1 User Based & Item Based Collaborative Filtering

User based and item based collaborative filtering are memory based techniques that we implemented in our recommender systems. In this section we will be talking about collaborative filtering using implicit ratings. User-based collaborative filtering finds recommended items for a user X by finding k similar users to user X using cosine similarity. It then estimates user X's rating for each item as a weighted average of the cosine similarity of the k similar users. Item based collaborative filtering finds recommended items for a user X by finding k similar items to each item using cosine similarity. It then estimates user X's rating for each item as a weighted average of the cosine similarity of the k similar items.

Cosine similarity is a measure of similarity between two users or items. It represents two users or items as vectors and calculates the cosine of the angle between the two vectors. Essentially, a user becomes more similar to another user the more times they have both used the same item. In item-based collaborative filtering, an item becomes more similar to another item the more times they have both been used by the same user. The resulting cosine is between 0 and 1, and the higher the value, the more similar the two users or items are.

In user-based collaborative filtering, the weighted average is calculated as follows: For each item, sum up the cosine similarities of the k similar users that have rated that item. Divide this sum by the sum of the cosine similarities of all of the k similar users to calculate a weighted average between 0 and 1. In item-based collaborative filtering, the weighted average is calculated using the k similar items instead of users. The higher the weighted average, the more likely the item will be recommended to user X. A recommender system can then sort all of the weighted averages from greatest to smallest and recommend the items in that order.

A memory-based approach is great for simple recommender systems because it is easy to implement and use. It does not require training or optimization like a model-based approach does, which means less time spent programming and no time spent training or optimizing. However, its simplicity does limit its performance. It does not perform as well when there is sparse data (Grover, 2017). Similarly, it can not work well with new users and items since there is no data for the recommender system to work off of (Koren, Bell, & Volinsky, 2009). These downsides limit its real world applications.

2.2.2 Matrix Factorization

Matrix Factorization is a model-based approach to collaborative filtering that we used to create a recommender system. Matrix factorization starts by separating data into two shallow matrices, one containing information about the users and one containing information about the items. Within the first shallow matrix, the data is going to be representative of user preferences broken into various categories. For example, if we were exploring a movie recommendation system, this matrix would list a user, and then include how much said user enjoys action movies, comedy movies, dramas, and more. These values are unique to each user, and dictate how much they are going to enjoy a particular item. The shallow matrix holding item data is representative of what qualities a particular item has in various categories. If we continue the previous example using movies, this matrix would hold a list of movies, and then include how much it can be considered an action movie, comedy movie, drama, and more.

Once we have both shallow matrices, we can estimate how much a particular user is going to enjoy an item by doing a dot product calculation between their data values and the data values of the item. If we repeat this process for all users and items, we eventually create a full matrix of values representing the ratings of each item by each user.

The difficulty arrives when we need to consider how the values in the shallow matrices are calculated. In a real world scenario, we would typically start with the data in the full matrix and then try and figure out what the values in the shallow matrices are. By using gradient descent and

additional machine learning models, we can tune the values in the shallow matrices until their dot product values are close enough to the real values in the full matrix. Once the shallow matrices consistently produce the best possible results in the full matrix, we may then use them to calculate new values that did not exist in the full matrix to begin with.

This model is the basis of how matrix factorization works; real data is used to tune specific values that are then in turn used to create new data. This new data is somewhat of an educated guess; however, we can test how accurate the prediction is using a testing set of data.

2.2.3 GAU Recommendation Model

The GAU Model is a joint model containing a **G**uardian-Guardian SPPMI matrix, **A**uxiliary information and a **U**RL-URL SPPMI matrix. It was developed by Nguyen Vo and Kyumin Lee (2019) as a recommender system that can recommend fact-checking URLs to Twitter users known as guardians. The GAU model hopes to provide personalized recommendations of fact-checking URLs to guardians so that they can then use them in their tweets.

A guardian shows interest in correcting fake news by replying to tweets with fact-checking URLs or retweeting the reply made by a guardian. Direct guardians are guardians that reply to tweets, and secondary guardians retweet the replies. Figure 2.2 shows a typical interaction between a Twitter user who posted an incorrect statement and guardians.

Figure 2.2: How Guardians Fact-Check on Twitter

The GAU model was compared with four other collaborative filtering algorithms: Bayesian Personalized Ranking Matrix Factorization, Matrix Factorization, CoFactor, and Collaborative Filtering Regression. The GAU model outperformed these algorithms in three different evaluation metrics. The objective function of the GAU model is shown in figure 2.3, which is from the research paper by Vo and Lee. Three of the most important variables in this equation are alpha_gau (α), beta_gau (β), and gamma_gau (γ). These parameters were important when we tuned the parameters of the GAU model.

$$\min_{\mathbf{U,V,L,K}} \mathcal{L}_{GAU} = \|\Omega \odot (\mathbf{X} - \mathbf{UV})\|_F^2 + \lambda(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2)$$
$$+ \|\mathbf{R}^{mask} \odot (\mathbf{R} - \mathbf{V}^T\mathbf{K})\|_F^2$$
$$+ \|\mathbf{G}^{mask} \odot (\mathbf{G} - \mathbf{UL})\|_F^2$$
$$+ \alpha \times \|\mathbf{S} - \mathbf{U}\mathbf{U}^T\|_F^2$$
$$+ \gamma \times Tr(\mathbf{U}^T\mathcal{L}_{uu}\mathbf{U})$$
$$+ \beta \times Tr(\mathbf{V}\mathcal{L}_{\ell\ell}\mathbf{V}^T)$$

Figure 2.3: Objective Function of the GAU model

2.2.4 Evaluating Recommender Systems

There are multiple metrics for evaluating the performance of a recommender system. Precision at k is the proportion of recommended items in the top k recommendations that are relevant (Malaeb, 2017). Common values of k are 10, 20, 50, and 100. Recall at k is the proportion of relevant items found in the top k recommendations. Normalized discounted cumulative gain (nDCG) at k measures the usefulness of an item based on its position in the recommended items list and its relevance (Stanford University, 2013). Note that a relevant item is an item that has been rated by the selected user, and a recommended item is an item that has not been rated by the user but is recommended by the recommendation system. The relevance of an item is the rating the user gave it. If implicit ratings are used, then the relevance of an item is 1 if it has been used by the user and 0 if it has not. Because nDCG is one of the most popular

ranking metrics, it is simple to calculate, and the GAU model already calculates it, we used nDCG as the metric to compare our recommendation systems (Zygmunt, 2015).

3 Methodology

Our goal was developing recommendation systems that can recommend fact-checking websites to guardians and building a web application that facilitates use of the recommendation systems. Our goal consisted of three objectives:

1. Comparing the results of User-Based, Item-Based, Matrix Factorization, and GAU recommendation systems;
2. Improving the GAU Model;
3. Building a web application to facilitate use of our recommendation systems.

To accomplish our objectives, we completed four tasks:

1. Implementing User-Based and Item-Based recommendation systems
2. Implementing a matrix factorization recommendation system
3. Improving the GAU model
4. Developing the web application

This chapter will go into our tasks in more detail and explain how they relate to our objectives.

3.1 Task 1. Creating User-Based and Item-Based Recommendation Systems

A great way to evaluate a recommendation system is by comparing it to others. For this reason, we decided to create basic recommendation systems as a basis for our comparison. User-Based

and Item-Based recommendation systems are simple recommendation systems, so we developed them first.

3.1.1 User-Based Recommendation System

One of the first recommendation systems we developed utilized User-Based collaborative filtering. User-Based collaborative filtering finds recommended items for a user X by finding k similar users to user X using cosine similarity and then estimates user X's rating for each item as a weighted average of the cosine similarity of the k similar users.

First, we created a 2D utility matrix of the data, where each row represented a user, each column represented a URL, and a 1 signified that the user of that row tweeted the url of that column. Then we split the data containing information about user X's tweet into a training, validation, and test dataset with a 80-10-10 split. The training dataset is used to train the model, the validation dataset is used to evaluate the set in order to tune the model's hyperparameters, and the test dataset is used to evaluate the model once tuning of the model is done. The results of the test dataset evaluation determine the effectiveness of the model. Because collaborative filtering is a simple recommendation system, it does not have any hyperparameters to tune, so we only used the training dataset and test dataset.

Next, we determined user X's similar users. To determine similar users, we used a metric called cosine similarity, which represents two users as vectors and calculates the cosine of the angle between the two vectors. Essentially, a user becomes more similar to another user the more times they have both linked the same URL in their tweets. We used scikit-learn, a machine learning

Python library, and their cosine_similarity function to do the calculations. Each user then has a cosine similarity value from 0 to 1 measuring their similarity to user X, where a greater value means a more similar user.

Third, we were able to estimate user X's rating for each URL as a weighted average of the cosine similarity of the k similar users. For each URL, we summed up the cosine similarities of the k similar users that have tweeted about that URL. We then divided that sum by the sum of the cosine similarities of all of the k similar users to calculate a weighted average between 0 and 1. The higher the weighted average, the more likely the URL will be recommended to user X.

Finally, we evaluated User-Based collaborative filtering by calculating precision, recall, and nDCG at k = 10, 20, 50, and 100. Precision at k is the proportion of recommended items in the top k recommendations that are relevant (Malaeb, 2017). Recall at k is the proportion of relevant items found in the top k recommendations. nDCG at k measures the usefulness of an item based on its position in the recommended items list and its relevance (Stanford University, 2013). Note that a relevant URL is a URL that has been tweeted by the selected user, and a recommended URL is a URL that has not been tweeted but is recommended by the recommendation system. We specifically used nDCG to compare the four different recommendation systems, which helped achieve our objective of comparing our recommendation models.

3.1.2 Item-Based Recommendation System

The other approach to collaborative filtering is Item-Based collaborative filtering. Instead of finding similar users to a user X, Item-Based collaborative filtering finds recommended items for user X by using cosine similarity to find k similar items to an item user X has not used and then estimates user X's rating for that item as a weighted average of the cosine similarity of the k similar items. Since Item-Based collaborative filtering is very similar to User-Based collaborative filtering (explained in section 3.1.1), we will only go into detail about the differences between the two in this section.

The utility matrix for Item-Based collaborative filtering represented users as columns and URLs as rows. After creating the training, validation, and test matrices, we used cosine similarity to determine the k most similar URLs to each URL for user X. A URL becomes more similar to another URL the more times they have both been tweeted by the same user. Then, we estimated user X's rating for each URL as a weighted average of the cosine similarity of the k similar URLs. For each URL, we summed up the cosine similarities of the k similar URLs that have also been tweeted by user X. We then divided that sum by the sum of the cosine similarities of all of the k similar URLs to calculate a weighted average between 0 and 1. The higher the weighted average, the more likely user X will like the URL and use it in a future tweet. Finally we evaluated Item-Based collaborative filtering by calculating precision, recall, and nDCG at k = 10, 20, 50, and 100. This evaluation helped achieve our objective of comparing the four recommendation systems.

3.2 Task 2. Creating Matrix Factorization Recommendation System

A more advanced collaborative filtering method, and the technique that the GAU model is built on, is matrix factorization. To get another point of comparison and better understand the GAU model, we developed a matrix factorization recommendation system.

We used Spotlight's implicit factorization model to implement our recommendation system (Kula, 2017). Spotlight uses PyTorch, an open source machine learning framework, to build its recommender models. In the paper "Matrix Factorization Techniques for Recommender Systems", Koren, Bell, and Volinsky (2009) describe matrix factorization as characterizing "both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to a recommendation." We used an implicit factorization model instead of an explicit factorization model because implicit ratings (what people choose to interact with) are widely accepted to be more meaningful than explicit ratings (ratings people explicitly give) (Kula, 2017).

The implicit factorization model has many parameters that affect how the model runs. We decided on the values of these parameters by testing the model multiple times with a wide variety of values, calculating the results with precision, recall, and ndcg value, and then comparing the tests to find the optimal set of parameters. The parameters we focused our tests on were loss function, number of embedding dimensions, number of iterations, batch_size, L2 loss penalty,

learning rate, and number of negative samples. After discovering the optimal parameters and calculating the precision, recall, and ndcg at k = 10, 20, 50, and 100, we made progress on our objective of comparing our four recommendation systems.

3.3 Task 3. Improving the GAU Model

The GAU model is the recommendation system developed by Nguyen Vo and Kyumin Lee (2019). It is built off of matrix factorization, and improves it by adding additional matrices and auxiliary information. Currently it outperforms User-Based collaborative filtering, Item-Based collaborative filtering, and matrix factorization. We tuned the parameters in order to improve it further.

We determined the optimal parameters for the GAU model by using a system of trial and error. This involved changing one parameter a small amount, training the model, and then comparing the results to previous tests. The initial parameters were single pointwise pointwise loss function and 0.8 for the value of alpha_gau, beta_gau, gamma_gau. The parameter with the largest impact on performance was the loss function, so first we determined which loss function (pointwise, bpr, hinge, or adaptive hinge) performed better. We then tuned the alpha_gau, beta_gau, and gamma_gau parameters, since these greatly impacted the results of the model. Since the other parameters did not make a noticable difference, we did not adjust them. To measure the performance of the model, we split the dataset into train, validation, and testing sets and then calculated the test ndcg at k = 10. The set of parameters that consistently produced the highest test ndcg at k = 10 was the optimal set. By performing these tests, we achieved our objective of

improving the GAU model and by calculating the ndcg, we partly achieved our objective of comparing the four recommendation systems.

3.4 Task 4. Developing the Web Application

The Web Application is the visual element of our project that allows users to see the results of the various recommendation systems online. The application is built using the MERN development stack, which stands for MongoDB, Express, React, and Node.js. Using Node.js, we built a back-end server using Express. This server would handle data collection and API interaction. MongoDB is a great NoSQL distributed database system that works great for storing our user information. React is a great JavaScript library that allows the creation of UI views using multiple different components.

Using the Node package “Concurrently”, the back-end Express server runs simultaneously with the front-end React Application. The Express server calls on multiple different routes when we need to fetch data. For the user data, we call on routes that access MongoDB. These routes accept a particular username, and then send the proper JSON response to the front-end.

The URL recommendation models require a lot of computational power, especially when there are over 12,000 URLs and almost 5,000 users in the dataset. Thus, we used AWS (Amazon Web Services) as our cloud computing solution, employing multiple EC2 virtual Linux machines to run scripts that insert the data into the database. In addition to the MongoDB serving data to the

application, we also employed the Twitter API to fetch real-time twitter information about a given user. To use the Twitter API, we had to register with Twitter to get the appropriate Auth Keys. Once we acquired those, we pass the desired route a Twitter username, and the API returns the data that we desire.

The front-end of the application is built using React, and to add extra appeal and ease of use, we used the React UI Framework Material-UI. Material-UI is a great framework developed by Google that allows easy integration of components. Within the React front end, we employed multiple react libraries to help present our data.

4 Results

After developing the user-based, item-based, and matrix-factorization recommendation systems, improving the GAU recommendation system, and evaluating all of them, we were able to improve the GAU model and found that it produces the best results out of all of the recommendation systems. We also created a web application that allows guardians to view the recommendations from all of the recommendation systems and other related data. In the following sections, we go into more depth on our findings.

4.1 Comparing the Results of User-Based, Item-Based, Matrix Factorization, and GAU Recommendation Systems

After developing the user-based, item-based, and matrix-factorization recommendation systems and improving the GAU recommendation system, we determined that the GAU recommendation system produces the most relevant recommendations and is the recommendation system that should be used to recommend fact-checking URLs to guardians.

We used normalized discounted cumulative gain (nDCG) as our evaluation metric. We describe our reasoning in section 2.3.4. Specifically, we calculated nDCG at k = 10 for each guardian and then found the average. Figure 4.1 shows a graph of the average nDCGs at k = 10 for all of the recommendation systems. The nDCG of the GAU model was 39% greater than the second highest nDCG from matrix factorization. Note that the user-based and URL-based

recommendation system results vary whenever a new training, testing, and validation set of matrices is generated, but stay the same when run multiple times using the same matrices. The matrix factorization and GAU model results change slightly every time they are trained, even if the parameters stay the same. The average nDCG at k = 10 shown for these two recommendation systems are the highest nDCGs we recorded.

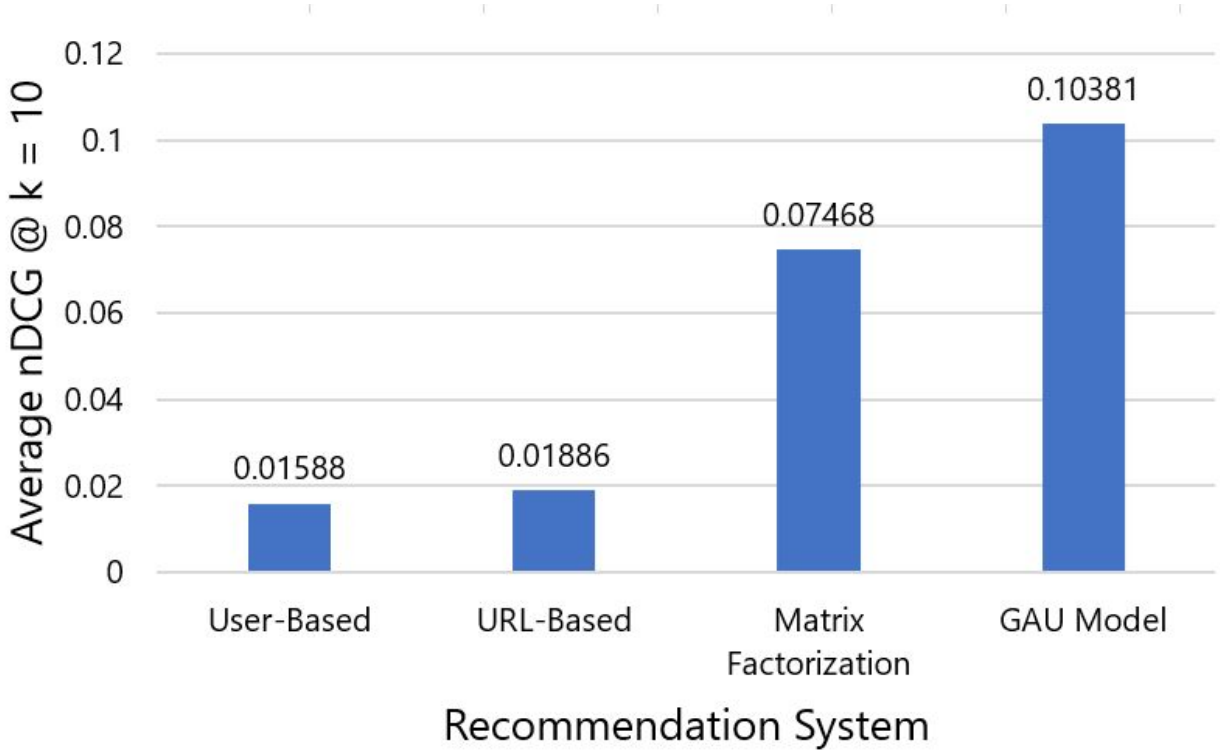

Figure 4.1: Average nDCG @ k = 10 for each Recommendation System

4.2 Improving the GAU Model

After tuning the parameters of the GAU model, we were able to improve its performance. We were able to achieve a test nDCG @ k = 10 14.5% greater than the previous max nDCG (Vo & Lee, 2019). We first tested the different loss functions: bpr, single pointwise square, adaptive

hinge, and hinge. These initial tests showed that hinge loss performed better than the other loss functions, which can be seen in Figure 4.2.

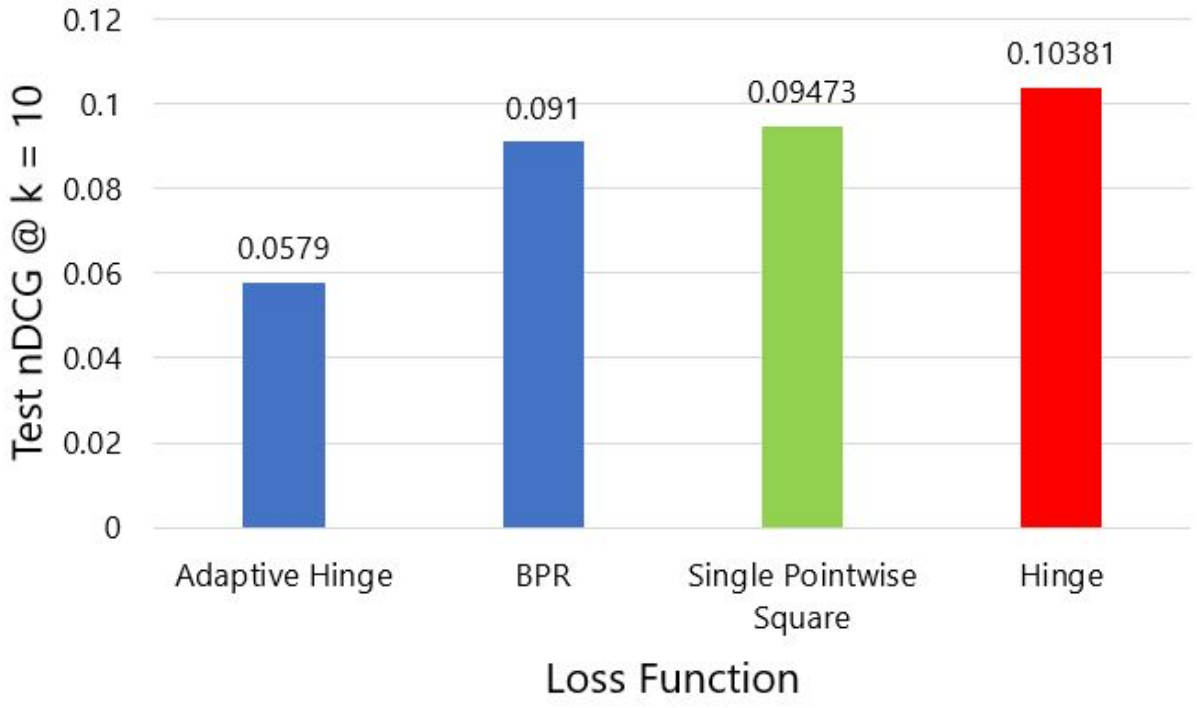

Figure 4.2: Best Test nDCG @ k = 10 for Loss Functions in GAU Model

We then tweaked the alpha_gau, beta_gau, and gamma_gau parameters. We found that keeping beta_gau and gamma_gau at 0 and setting alpha_gau to a large value achieved the best results for hinge loss. We found that 2.1 to 2.5 was where the nDCG was the greatest, so we tested the alpha_gau in this range in increments of 0.01. Each value of alpha_gau was tested at least 3 times, since the results varied each time. We then recorded the highest nDCG out of the multiple tests for each alpha_gau value. All of our tests showed that the nDCG was highest when alpha_gau was equal to 2.28. Figure 4.3 shows a graph with the highest test nDCGs at k = 10 for values of alpha_gau between 2.1 and 2.5.

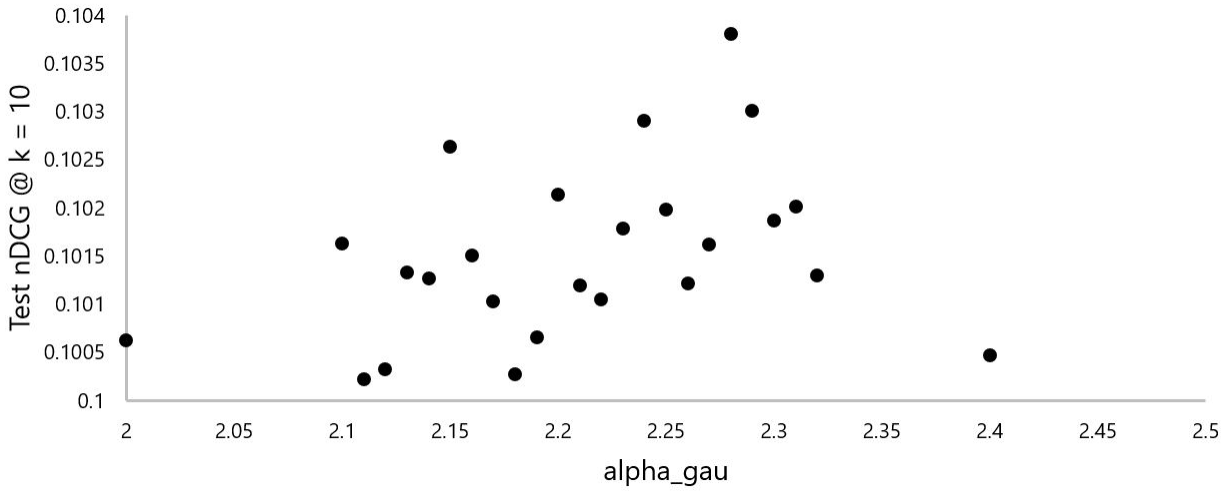

Figure 4.3: Best Test nDCG @ k = 10 for Different Values of alpha_gau with Hinge Loss

4.3 Building a Web Application to Facilitate Use of our Recommendation Systems

The web application is currently operational, and can be accessed at https://floating-retreat-14391.herokuapp.com/. Broken down into three major sections, the web application opens on the search view, introducing the application with a title image, and prompting the user to enter a valid Twitter username. Figure 4.4 is an image of the search view.

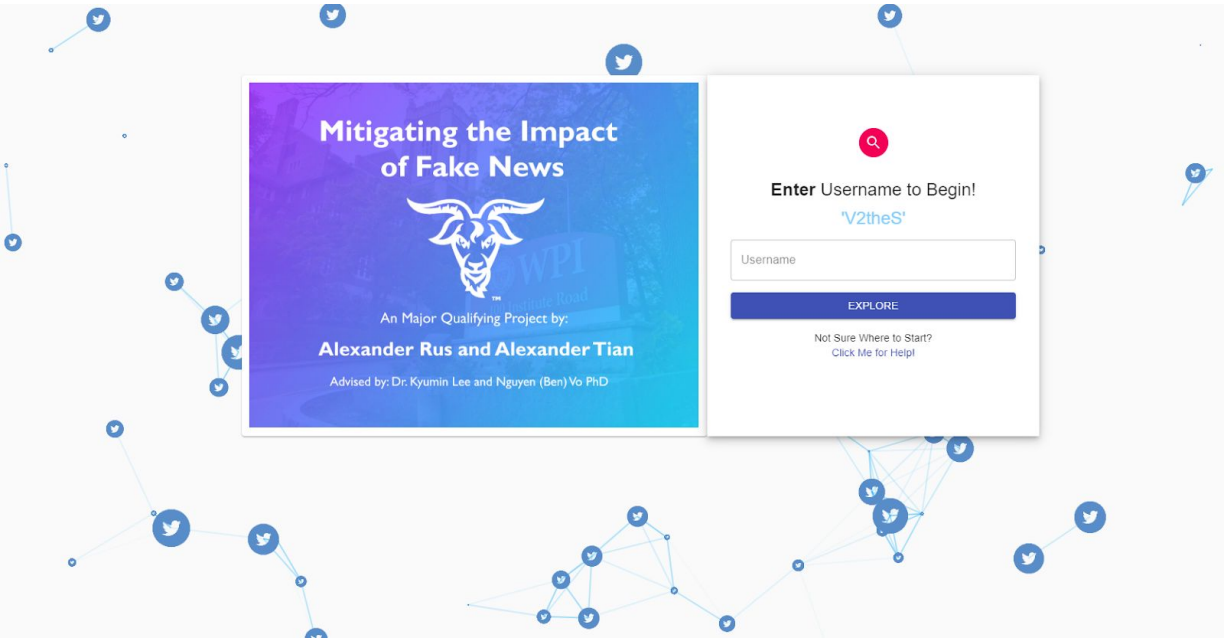

Figure 4.4: The search view introduces the project and prompts for a Twitter username

Once the user enters a Twitter username, they will be brought to the main page. The main page is broken down into two major sections, one labeled “Results” and one labeled “About”. The “Results” section holds the Twitter user information collected using the Twitter API, as well as the lists of recommended URLs based on the four different recommendation models. Should a user click on one of these options, the main page will display the appropriate results. Figure 4.5 displays the recommended URLs for a specific user using the GAU recommendation model.

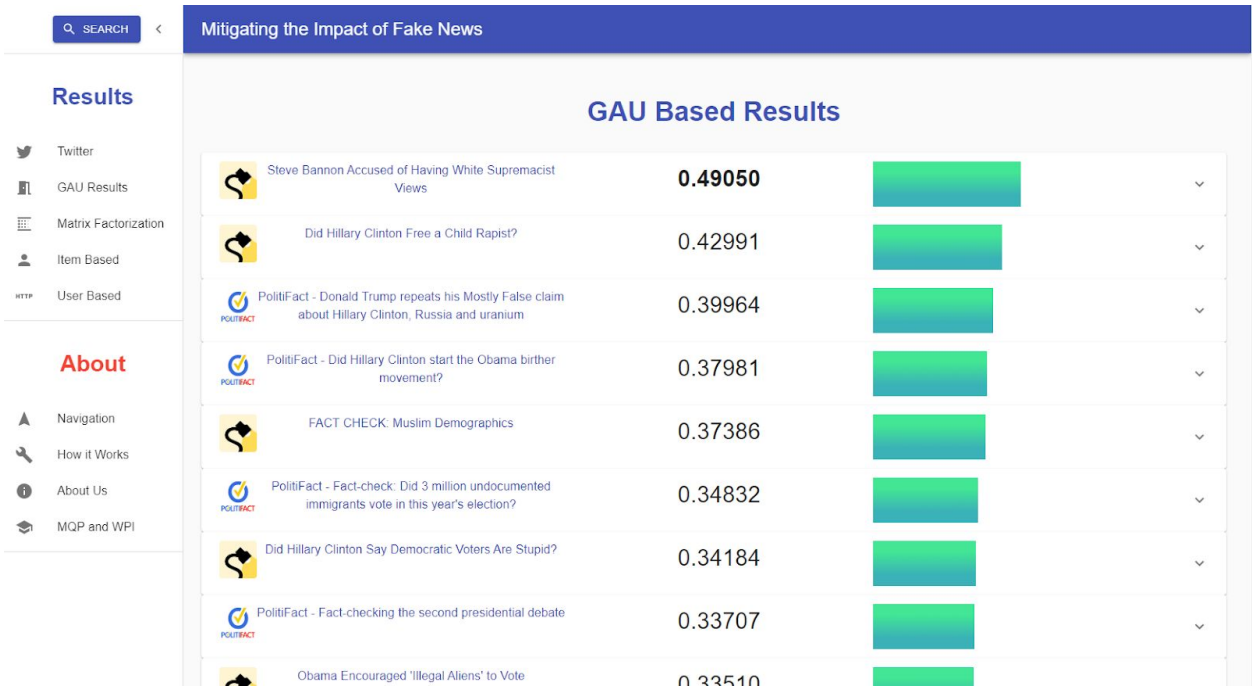

Figure 4.5: The Recommended URLs for a given user using the GAU model

Beyond the “Results'” section, the “About” section is intended to educate the user about the web application and how we arrived at the given results. The “Navigation” option displays how one may navigate the web application, the “How it Works” option explains the the logic behind the four different recommendation models, the “About Us” option provides information about the MQP team behind the project, and the “MQP and WPI” option gives some basic information about Worcester Polytechnic Institute as well as the MQP process.

The goal behind the web application was to allow the public to visually see the results of the recommendation systems in a responsive and intuitive manner. The web application achieves this goal through its uses of numerous interfaces working concurrently to provide information rapidly and in a visually appealing way.

5 Conclusions and Future Work

We created three recommendation systems, improved the GAU model, compared all of the recommendation systems, and then built a web application to facilitate use of our recommendation models. After drawing conclusions from our work, we came up with recommendations for future work to make the GAU model and web application ready for public use. The following sections describe the conclusions and recommendations we made.

5.1 Conclusions

We compared four recommendation systems: user-based collaborative filtering, item-based collaborative filtering, matrix factorization, and the GAU model. When comparing the best nDCG at k = 10 of each of the recommendation systems, we found that the GAU model performs the best. The matrix factorization recommender system is second best, and the other recommender systems perform significantly worse.

We tuned the parameters of the GAU model to try to improve its performance. Our work improved the GAU model by 14.5% when comparing the best test nDCG at k = 10 of the model before and after tuning. This was largely achieved by using the hinge loss function instead of single pointwise square loss, setting the alpha_gau parameter to 2.28, and setting the beta_gau and gamma_gau parameters to 0.

5.2 Future Work

The goal of our project and the GAU model is to recommend fact-checking URLs to guardians. However, there is still future work that must be done before that goal is achieved.

1. The web application can be improved to be more efficient and allow recommendation to happen dynamically rather than statically. Instead of providing the results from the users provided in the dataset, we could allow new users to have their recommended URLs calculated in real time.
2. The GAU model's parameters can be tuned further. Due to time constraints, we were not able to test as many combinations of parameters that we would have liked to. We did most of our tests with the hinge loss function, as we got the best results from it. There may be a way to improve the GAU model further with different loss functions, such as bpr, single pointwise square, and adaptive hinge, that we did not test as extensively.
3. Improvements can be made to the GAU model. As stated in Vo's and Lee's (2019) research paper on the GAU model, there are some potential improvements that can be made in future work. One improvement is to address the cold-start issue with guardians that posted less than three fact-checking URLs. Another potential improvement is to look into deep learning techniques and if they could further improve the performance of the model.

References

Clement, J. (2020, February). Social media: global penetration rate 2020, by region. *Statista.* Retrieved from https://www.statista.com/statistics/269615/social-network-penetration-by-region/

Grover, H. (2017, December). Various Implementations of Collaborative Filtering. *Towards Data Science.* Retrieved from https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0

Hameleers, M. & van der Meer, T. G. L. A. (2019, January). Misinformation and Polarization in a High-Choice Media Environment: How Effective Are Political Fact-Checkers?. *Sage Journals, 47*(2), 227-250. doi: 10.1177/0093650218819671

Hindman, M. & Barash, V. (2018, October). Disinformation, 'Fake News' and Influence Campaigns on Twitter. *Knight Foundation.* Retrieved from: https://knightfoundation.org/reports/disinformation-fake-news-and-influence-campaigns-on-twitter/

Koren, Y., Bell, R., & Volinsky, C. (2009, August). Matrix Factorization Techniques for Recommender Systems. *IEEE.* 42(8), 30-37. doi: 10.1109/MC.2009.263

Kula, M. (2017). *Spotlight*. Retrieved from https://github.com/maciejkula/spotlight

Liao, Kevin. (2018, November). Prototyping a Recommender System Step by Step Part 1: KNN Item-Based Collaborative Filtering. *Towards Data Science.* Retrieved from https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea

Malaeb, M. (2017). *Recall and Precision at k for Recommender Systems.* Retrieved from https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. *Springer.* 1-35. Retrieved from http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf

Shearer, E. & Matsa, K. E. (2018, September). News Use Across Social Media Platforms 2018. *Pew Research Center.* Retrieved from https://www.journalism.org/2018/09/10/news-use-across-social-media-platforms-2018/

Soll, J. (2016, December). The Long and Brutal History of Fake News. *Politico Magazine*. Retrieved from https://www.politico.com/magazine/story/2016/12/fake-news-history-long-violent-214535

Stanford University. (2013). "Introduction to Information Retrieval - Evaluation" (PDF). Retrieved from http://www.stanford.edu/class/cs276/handouts/EvaluationNew-handout-6-per.pdf

Stroud, N. J. (2007, December). Media Use and Political Predispositions: Revisiting the Concept of Selective Exposure. *Political Behavior*, *30*(3), 341-366. doi: 10.1007/s11109-007-9050-9

Vo, N. & Lee, K. (2019, August). The Rise of Guardians: Fact-checking URL Recommendation to Combat Fake News. *WPI Computer Science Department.* Retrieved from https://arxiv.org/pdf/1806.07516.pdf

Zygmunt, Z. (2015, August). Evaluating recommender systems. *FastML.* Retrieved from http://fastml.com/evaluating-recommender-systems/

Appendix: MQP Description and Relevance

WPI has a unique curriculum that pushes the idea of project based learning. Students are motivated to look for project opportunities both on and off campus. The Major Qualifying Project (MQP) is a large scale project that assesses students' knowledge in their specific field of study. Completed in the senior year of undergraduate study, the MQP often involves a combination of research and design. Students, either independently or in teams, work under one or more professors who advise them in the project process. Our project, Mitigating the Impact of Fake News, is an MQP that took place on WPI's campus. We worked on this project over a three-term period, which started on August 23, 2019, and we presented the project on April 24, 2020.